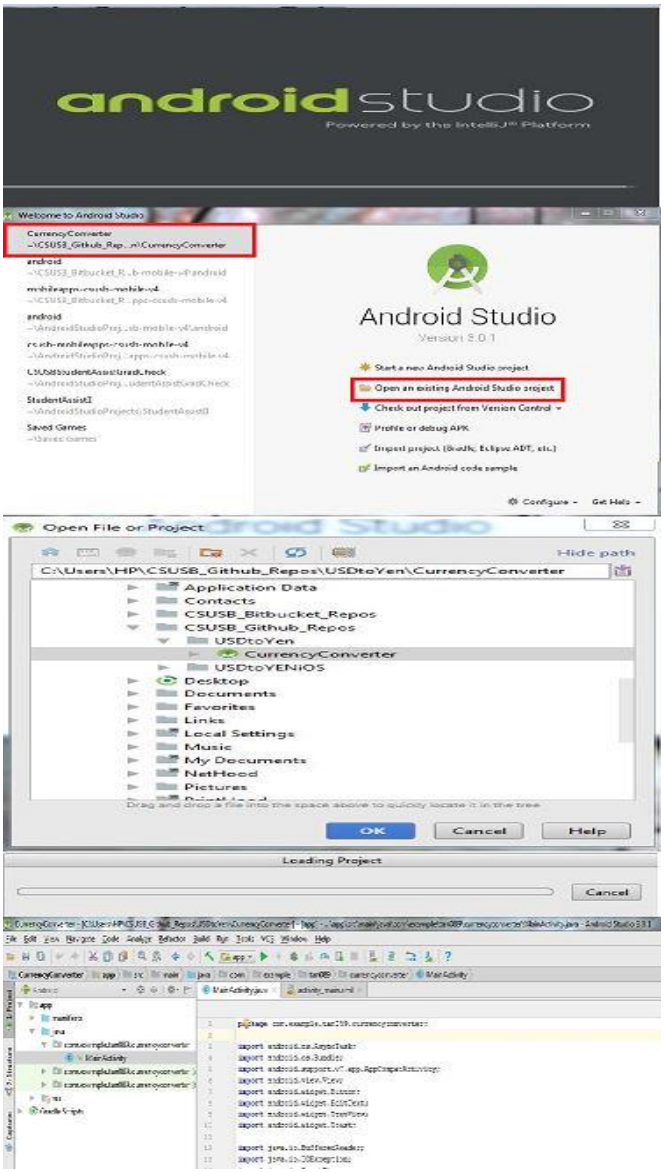
App: US Dollar to Japanese Yen Converter

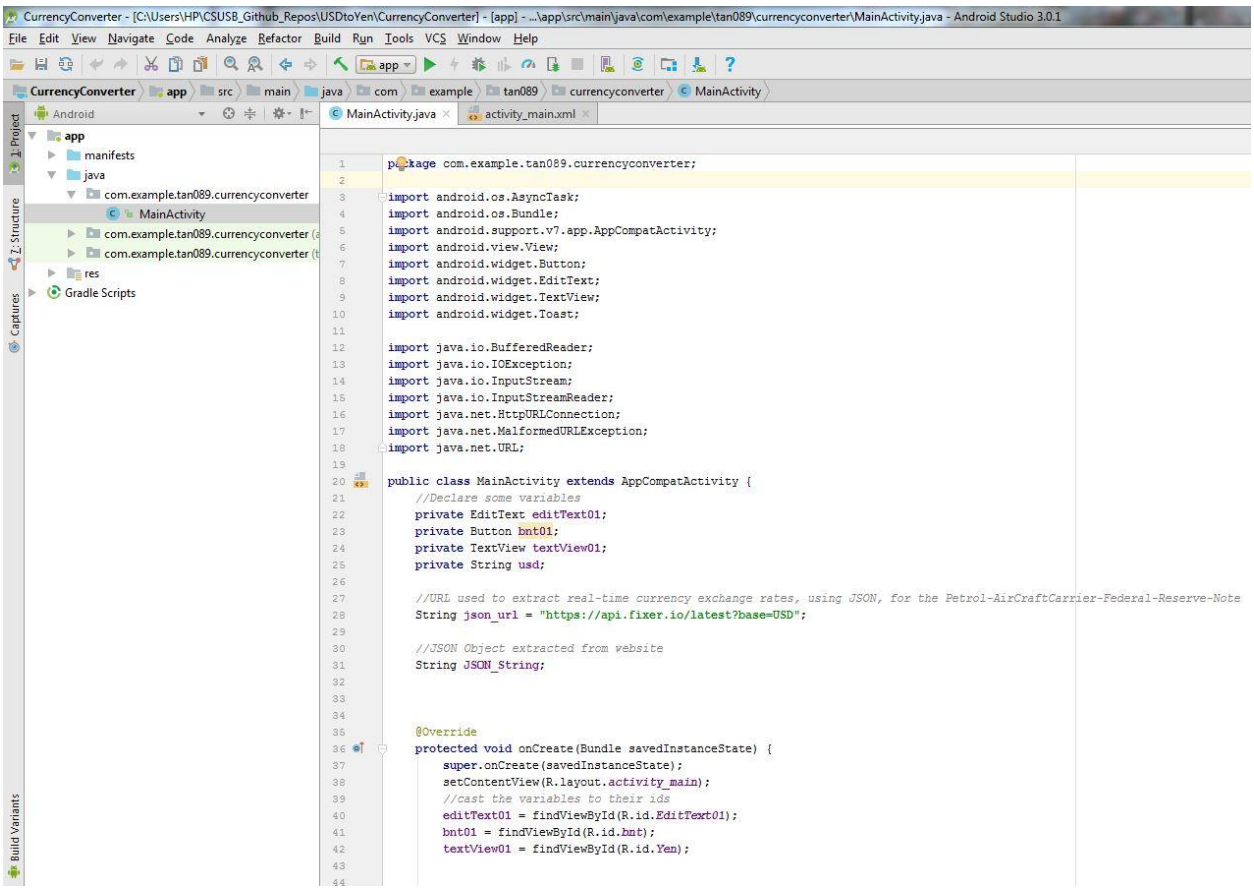
NOTE: The Volley Library can be used in place of the code in this tutorial. The output of the currency conversion must be the same, however.

1. Open Android Studio, open app project “CurrencyConverter” from Week 1: Android Studio Basic Tutorial.



2. Navigate to, and open the file MainActivity.java in the path: …\USDtoYen\CurrencyConverter\app\src\main\java\com\example\your_user_name\currencyconverter**MainActivity.java**

   Open **MainActivity.java** in Android Studio.

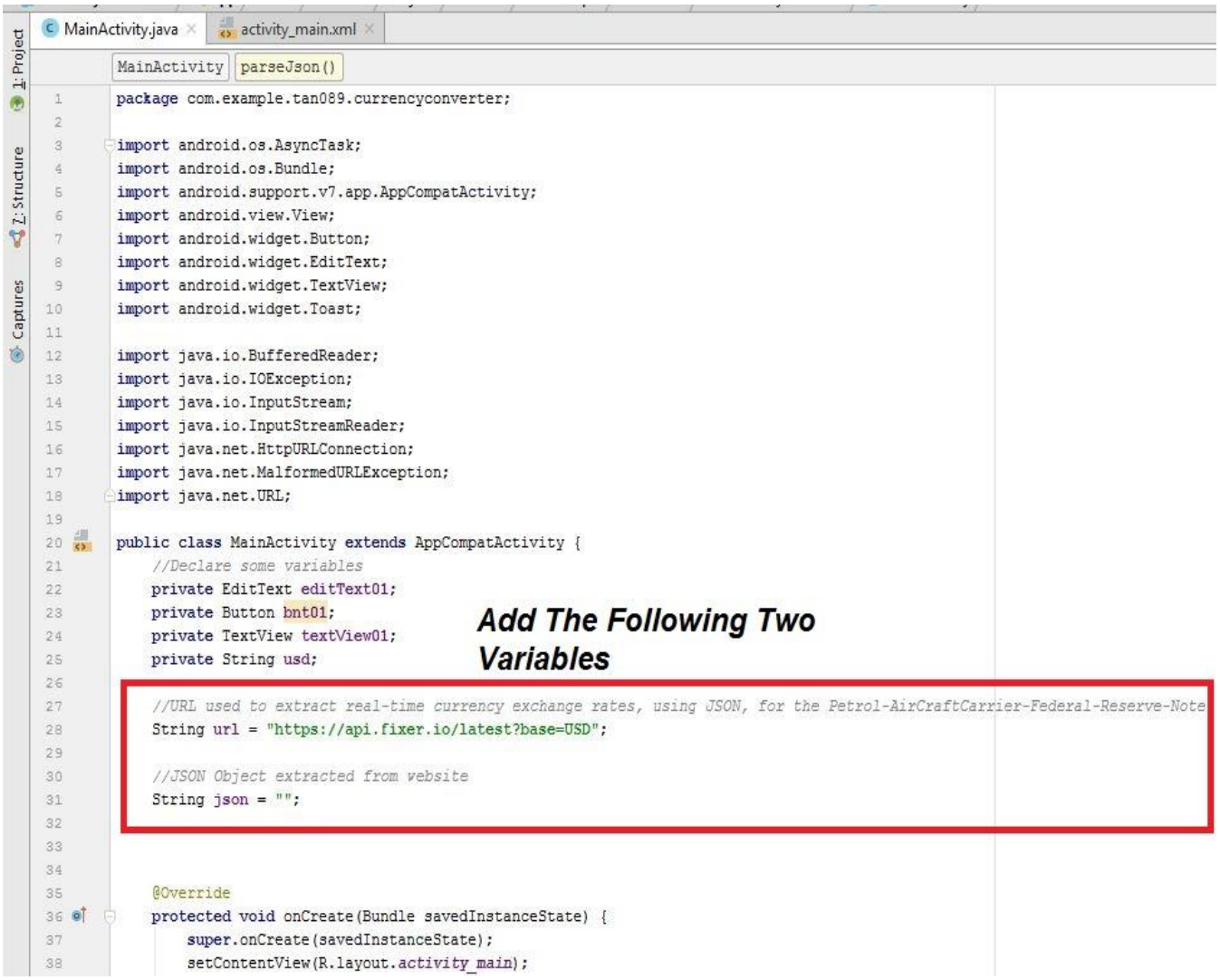


3. Let us begin. Move the cursor into the top of the ***MainActivity*** Class and declare two new variables of the String Class, below the variable **usd** but above the **onCreate(…)** method.

   These two variables are **url** and **json**.
   The **url** variable will store the URL for the website that we are using to extract a JSON object. (JSON is an acronym that stands for Java Script Object Notation)
   For this tutorial, the URL we are using is https://api.fixer.io/latest?base=USD

The **url** variable will be initialized to this URL in the form of a String.

The second variable, **json**, will be initialized as an empty string. We will use the string **json** later within our code, in order to store the JSON object that will be extracted from the website. This extraction process will be done within the **Asynchronous Task**. (The JSON object extracted will be in the form of a string and it will be the most up to date currency conversion value for USD (U.S. Dollars).




```
package com.example.tan089.currencyconverter;

import android.os.AsyncTask;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.MalformedURLException;
import java.net.URL;

public class MainActivity extends AppCompatActivity {
    //Declare some variables
    private EditText editText01;
    private Button bnt01;
    private TextView textView01;
    private String usd;

    //URL used to extract real-time currency exchange rates, using JSON, for the Petrol-AirCraftCarrier-Federal-Reserve-Note
    String url = "https://api.fixer.io/latest?base=USD";

    //JSON Object extracted from website
    String json = "";


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
```

4. In the Click Event method, **bnt01.setOnClickListener(…**,
   let us prepare by *converting* the String value in the variable **json** to a type Double.
   We will do this by declaring and initializing a variable of type Double, called **convert**.

   Convert the JSON string to type double and initialize the value of type double to the variable **convert**, all at once like so:

```
double convert = Double.parseDouble(json);
```

```
//Click event
bnt01.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View convertToYen) {
        //obtain the latest currency-conversion value from URL, convert from String to Double
        double convert = Double.parseDouble(json);

        //covert user's input to string
        usd = editText01.getText().toString();
        //if-else statement to make sure user cannot leave the EditText blank
        if (usd.equals("")){
            textView01.setText("This field cannot be blank!");
        } else {
            //Convert string to double
            Double dInputs = Double.parseDouble(usd);

            //Convert function, using the value obtained from the URL, multiply it by the user input
            Double result = dInputs * convert;
            //Display the result
            textView01.setText("$" + usd + " = " + "¥"+String.format("%.2f", result));
            //clear the edittext after clicking
            editText01.setText("");
        }
    }
});
}
```

Initialize variable, "convert", here

   The variable, **convert**, replaces the value of type double in the initialization of the variable, **result**, from the tutorial in Week 1.

   Let us recall that the variable **json** is an empty string as of now, but later, it will be assigned a value of type string from the website that will be accessed in the **Asynchronous Task**. This step is in preparation for when we obtain this JSON object.

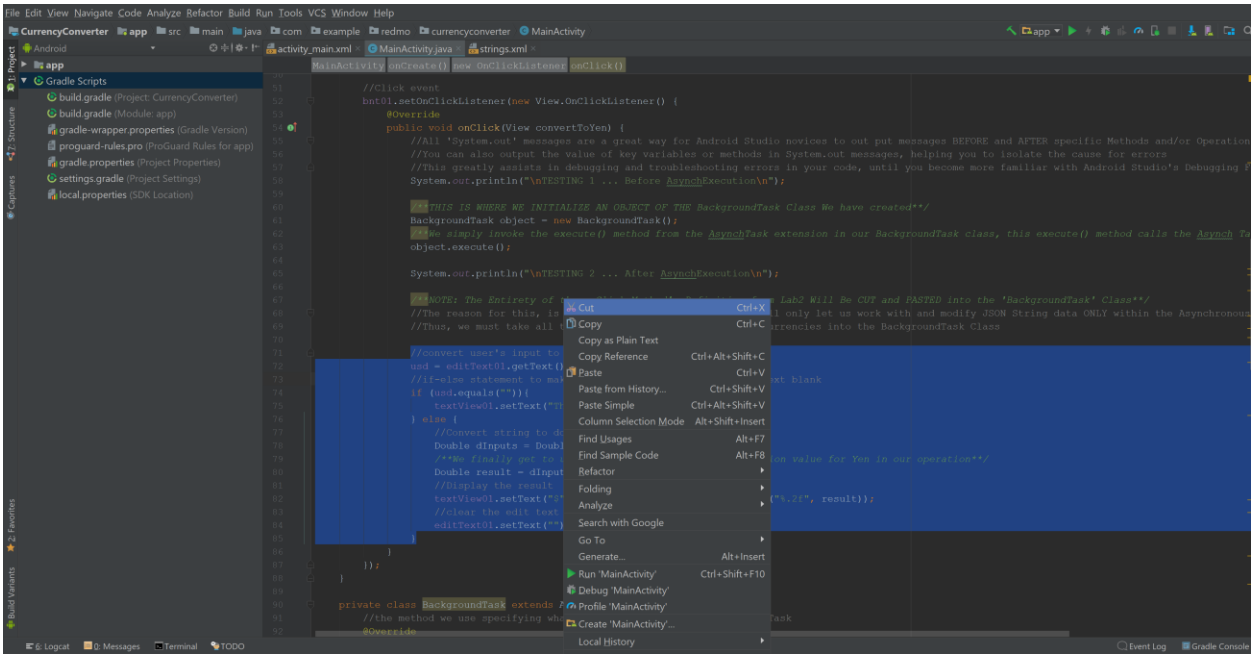
5. Now, we will begin to write the class that will contain the **<u>Asynchronous Task</u>** used to extract the JSON object from the currency-conversion web Application Programming Interface, api.fixer.io
   We will name the class ***BackgroundTask*** and write it below the Click Event method, **bnt01.setOnClickListener(…**

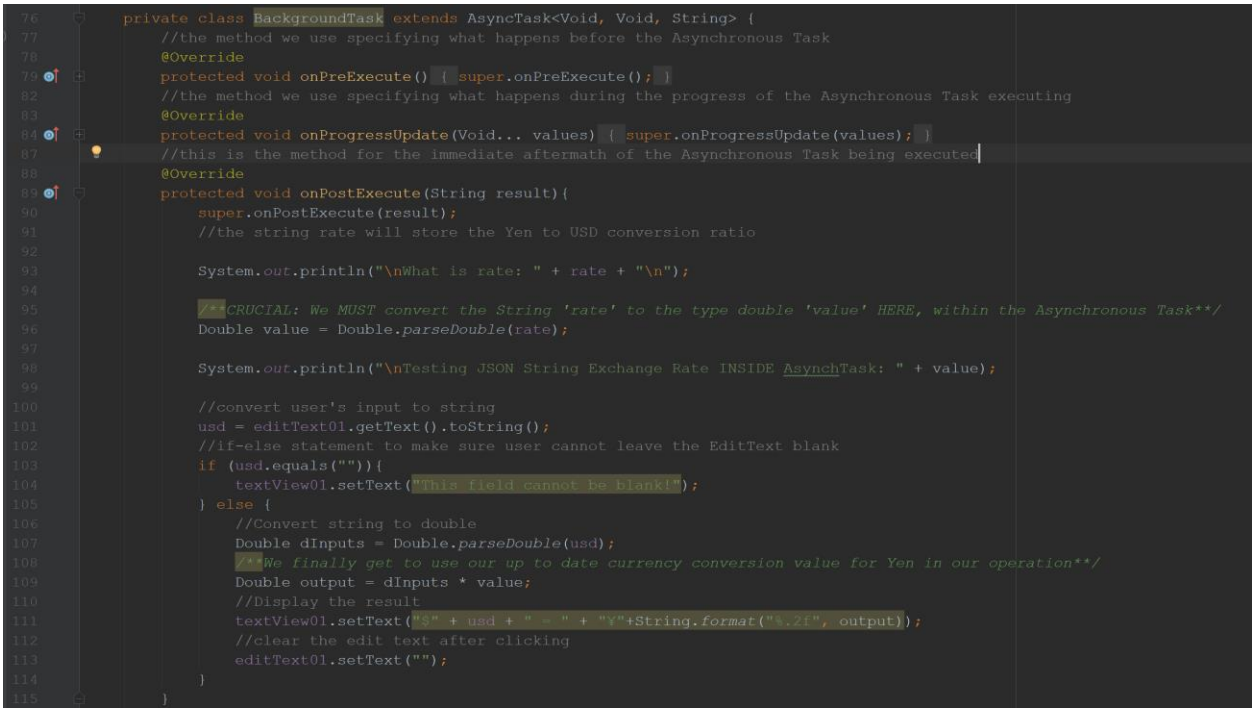
```
38          //Click event
39          bnt01.setOnClickListener(new View.OnClickListener() {
40              @Override
41              public void onClick(View convertToYen) {
42                  //obtain the latest currency-conversion value from URL, convert the JSON from String to Double
43                  double convert = Double.parseDouble(json);
44
45                  //covert user's input to string
46                  usd = editText01.getText().toString();
47                  //if-else statement to make sure user cannot leave the EditText blank
48                  if (usd.equals("")){
49                      textView01.setText("This field cannot be blank!");
50                  } else {
51                      //Convert string to double
52                      Double dInputs = Double.parseDouble(usd);
53
54                      //Convert function, using the value obtained from the URL, multiply it by the user input
55                      Double result = dInputs * convert;
56                      //Display the result
57                      textView01.setText("$" + usd + " = " + "¥"+String.format("%.2f", result));
58                      //clear the edittext after clicking
59                      editText01.setText("");
60                  }
61              }
62          });
63      }
64
65      private class BackgroundTask extends AsyncTask<String, Void, String> {
66          @Override
67          protected String doInBackground(String... params) {
68
69              //Stub: More code will define this nested method in the BackgroundTask class later
70              //The code written in this method will be the primary instructions carried out within the Asynchronous Task
71
72              return null;
73          }
74
75      }
```

Begin to write the <u>BackgroundTask</u> Class Below. The nested method, doInBackground(String... params) will be defined more later

The class, <u>BackgroundTask,</u> will contain most of the instructions for the app's Asynchronous Task

6. The logical code for currency conversion must be placed in the nested method, within the BackgroundTask Class, **onPostExecute(String result)**
   Take the logical code for currency conversion from Tan's code, then cut and paste it into the onPostExecute(String result) method



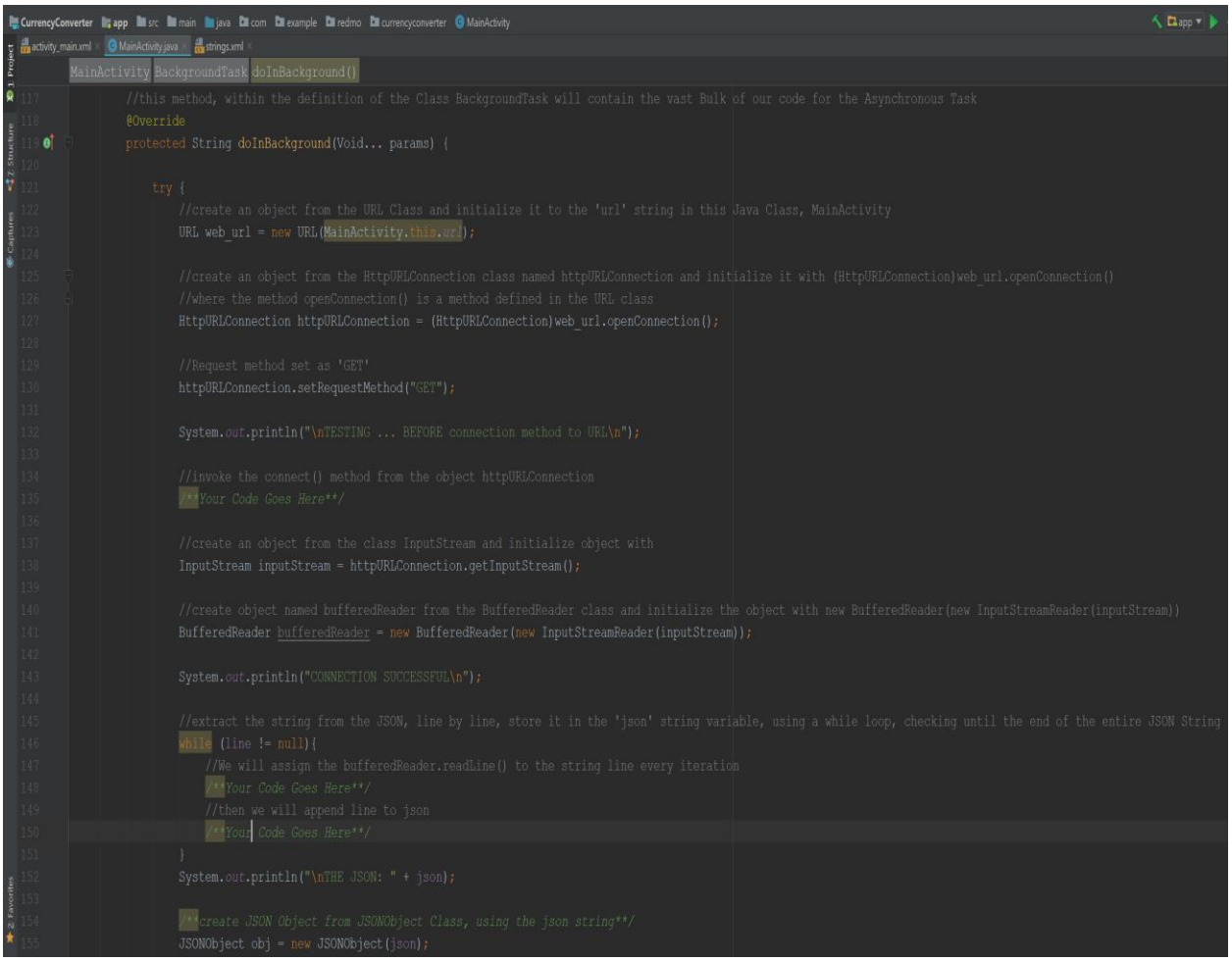


```
private class BackgroundTask extends AsyncTask<Void, Void, String> {
    //the method we use specifying what happens before the Asynchronous Task
    @Override
    protected void onPreExecute() { super.onPreExecute(); }
    //the method we use specifying what happens during the progress of the Asynchronous Task executing
    @Override
    protected void onProgressUpdate(Void... values) { super.onProgressUpdate(values); }
    //this is the method for the immediate aftermath of the Asynchronous Task being executed
    @Override
    protected void onPostExecute(String result){
        super.onPostExecute(result);
        //the string rate will store the Yen to USD conversion ratio

        System.out.println("\nWhat is rate: " + rate + "\n");

        /**CRUCIAL: We MUST convert the String 'rate' to the type double 'value' HERE, within the Asynchronous Task**/
        Double value = Double.parseDouble(rate);

        System.out.println("\nTesting JSON String Exchange Rate INSIDE AsynchTask: " + value);

        //convert user's input to string
        usd = editText01.getText().toString();
        //if-else statement to make sure user cannot leave the EditText blank
        if (usd.equals("")){
            textView01.setText("This field cannot be blank!");
        } else {
            //Convert string to double
            Double dInputs = Double.parseDouble(usd);
            /**We finally get to use our up to date currency conversion value for Yen in our operation**/
            Double output = dInputs * value;
            //Display the result
            textView01.setText("$" + usd + " = " + "¥"+String.format("%.2f", output));
            //clear the edit text after clicking
            editText01.setText("");
        }
    }
```

Change the name 'result' of the variable that is of type string to the name 'output'

7. The following screen shot shows the majority of code you will need for your BackgroundTask class, that extends AsyncTask. You will need to create objects from the following classes: URL, HttpURLConnection, InputStream, BufferedReader, and JSONObject.

   You will implement over several methods from the objects of these classes in order to retrieve the JSON, up-to-date currency conversion rate. Again, you are more than welcome to use the Volley library, supported by Android Studio, in place of using the verbose classes for the Asynchronous Task.



```
//this method, within the definition of the Class BackgroundTask will contain the vast Bulk of our code for the Asynchronous Task
@Override
protected String doInBackground(Void... params) {

    try {
        //create an object from the URL Class and initialize it to the 'url' string in this Java Class, MainActivity
        URL web_url = new URL(MainActivity.this.url);

        //create an object from the HttpURLConnection class named httpURLConnection and initialize it with (HttpURLConnection)web_url.openConnection()
        //where the method openConnection() is a method defined in the URL class
        HttpURLConnection httpURLConnection = (HttpURLConnection)web_url.openConnection();

        //Request method set as 'GET'
        httpURLConnection.setRequestMethod("GET");

        System.out.println("\nTESTING ... BEFORE connection method to URL\n");

        //invoke the connect() method from the object httpURLConnection
        /**Your Code Goes Here**/

        //create an object from the class InputStream and initialize object with
        InputStream inputStream = httpURLConnection.getInputStream();

        //create object named bufferedReader from the BufferedReader class and initialize the object with new BufferedReader(new InputStreamReader(inputStream))
        BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(inputStream));

        System.out.println("CONNECTION SUCCESSFUL\n");

        //extract the string from the JSON, line by line, store it in the 'json' string variable, using a while loop, checking until the end of the entire JSON String
        while (line != null){
            //We will assign the bufferedReader.readLine() to the string line every iteration
            /**Your Code Goes Here**/
            //then we will append line to json
            /**Your Code Goes Here**/
        }
        System.out.println("\nTHE JSON: " + json);

        /**create JSON Object from JSONObject Class, using the json string**/
        JSONObject obj = new JSONObject(json);
```

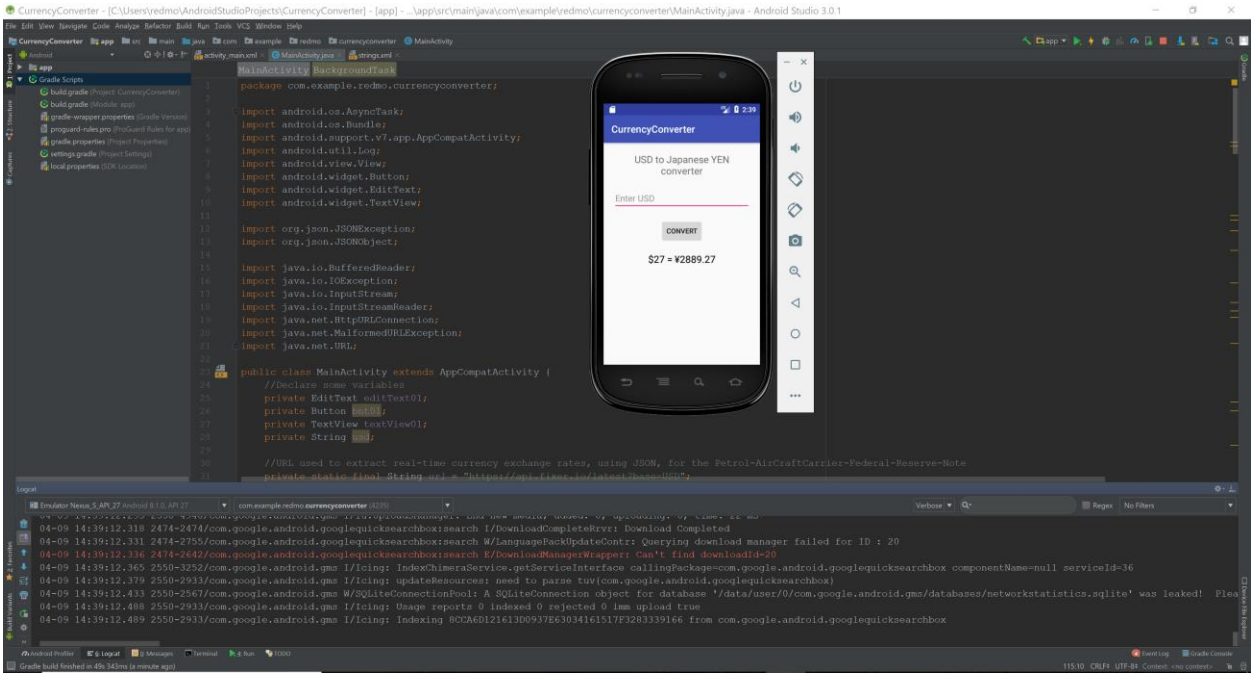
I leave merely 3 lines blank, for you to fill in your code. Each are very basic instantiations of methods called from objects and the basic use of the assignment operator. (I will go over these 3 lines in the Laboratory Tutorial);

8. The following screen shot shows what should be inside the end of the try, catch, and throw block of the BackgroundTask Class. Also, it shows how to use the JSONObject class to instantiate an object to acquire the web-page label data “rates”.

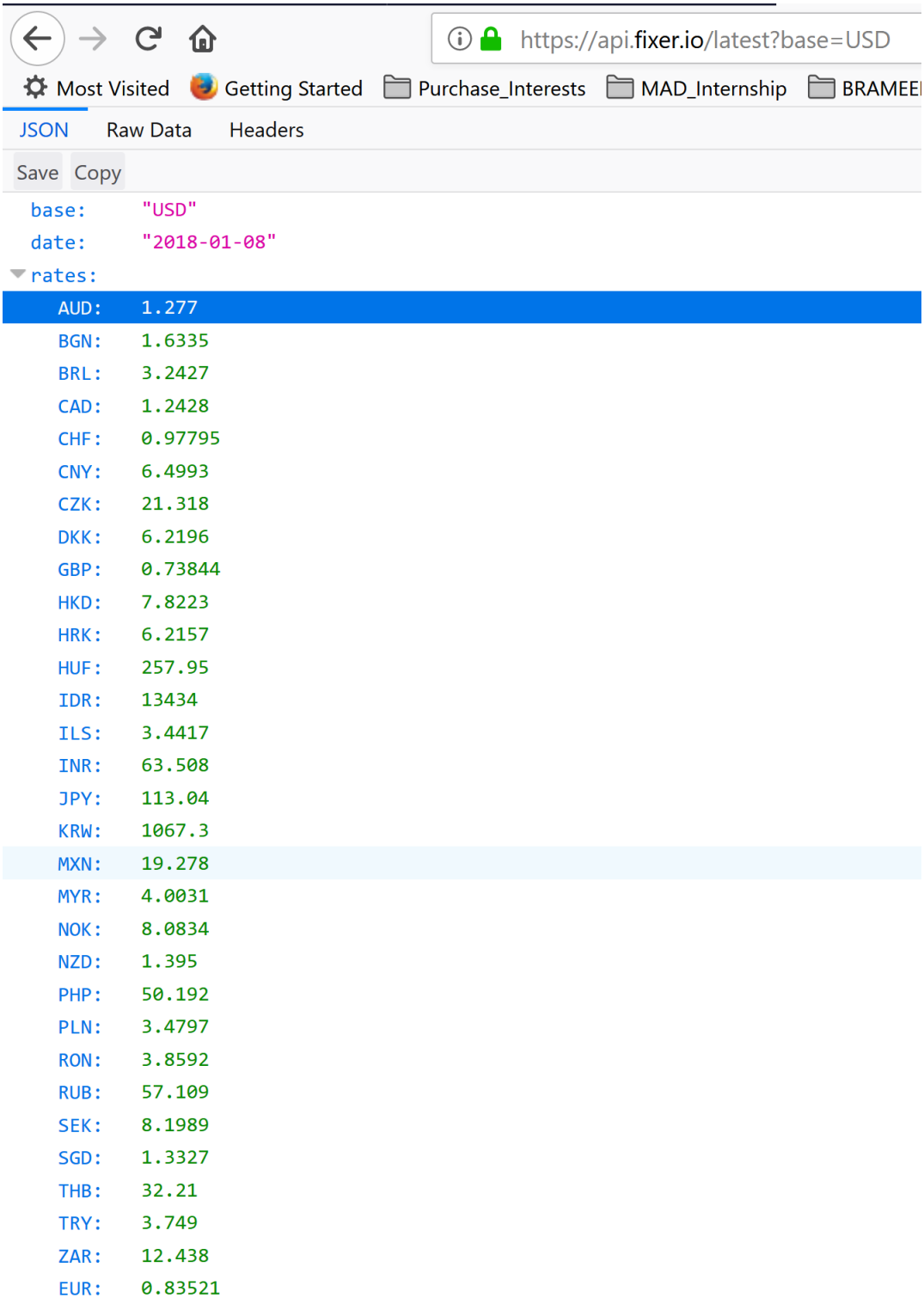
   In addition, it is necessary to convert this data to type String first, before using the data for any more tasks.

```
            //create second JSON Object that will contain a nested JSON Object within the FIRST JSON Object created
            JSONObject objRate = obj.getJSONObject("rates");

            //use the second JSON Object created and use the get(String ...) method
            //We will put "JPY" as the argument in the parameter of the get(String ...) method in order to get the exchange rate for Yen
            //The logic code for currency conversion MUST GO INTO the onPostExecute() Method within AsyncTask

            rate = objRate.get("JPY").toString();



        } catch (MalformedURLException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        } catch (JSONException e) {
            Log.e( tag: "MYAPP",  msg: "unexpected JSON exception", e);
            System.exit( status: 1);
        }
        return null;
    }

}
}
```

9. Finally, this snapshot shows what your emulator should look like after you have successfully implemented Asynchronous task.

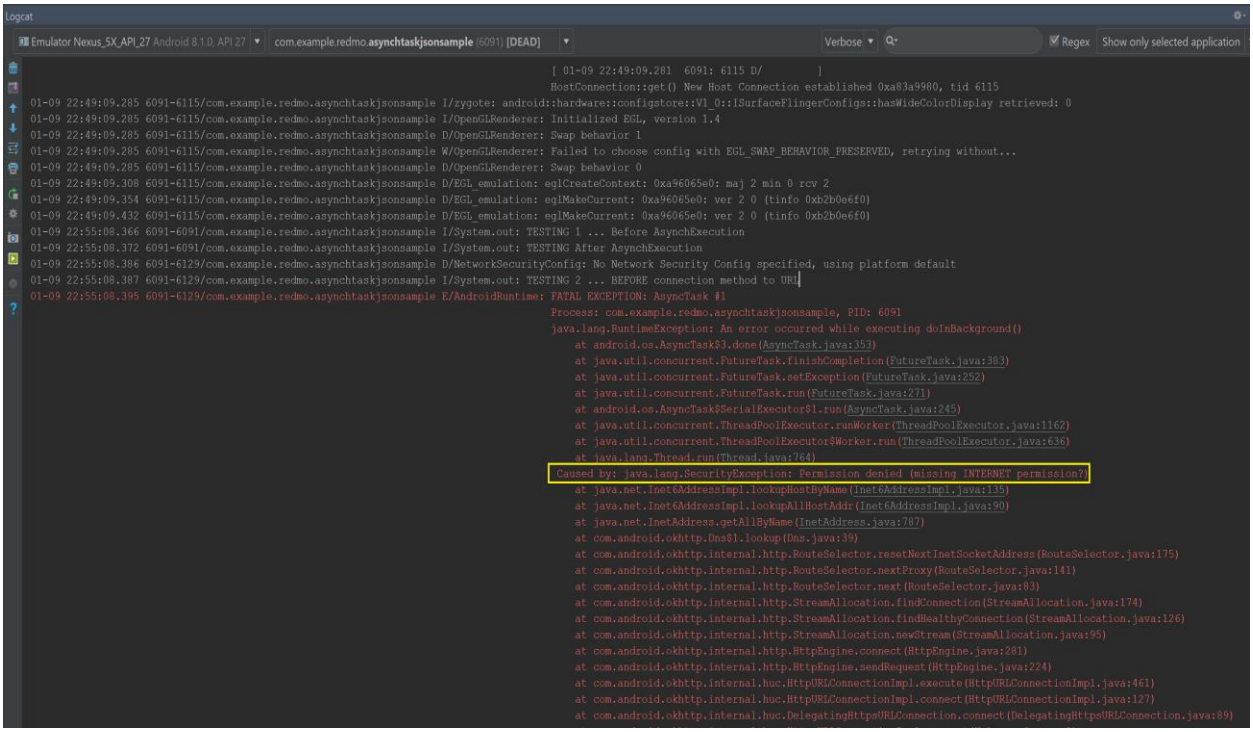


10. The following is URL JSON data from the website: api.fixer.io



11. The Missing Internet Permission

Depending on the configuration of your Manifest.xml file in Android Studio, you will likely not have the proper Internet Permission configured. Do not be surprised if you get this error the first time you build and run the app with Asynchronous Task:



To fix this error, you will need to go into you Manifest.xml file in Android Studio, and then copy-paste the proper permission code for your app to access the Internet and obtain the JSON data for the currency conversion.

You can get this .xml line for your Manifest file by simply doing a Google search with terms such as “Missing Internet Permission Android” and the answer will come right up. This is a very simple, common issue when doing beginner Android Studio programming, so it is very easy to find, thankfully.